Week 1: Assignment

1. How does the functionality of a design be specified in terms of its behavior?
    a. Boolean expression or truth table.
    b. Algorithms written in high level languages.
    c. State transition diagram/table.
    d. All of the above.

    Answer: (d)

    The functionality of a design at the behavioral level can be specified as Boolean equation, truth table, state transition diagram/table and algorithms written in some high level languages. Thus option (d) is correct.

2. Which of the following represents a structural representation for the function f=A.B+C?
    a. The truth table of the function with output column (01010111)
    b. A Verilog specification: assign f = (A & B) | C;
    c. A netlist consisting of one 2-input AND and one 2-input OR gate
    d. None of the above

    Answer: (c)

    The truth table and verilog "assign" statement represent design at the behavioral level. Thus option (a) and (b) are false. At structural level functionality can also be specified as a netlist consisting of logic gates. Thus option (c) is true.

3. Given a gate-level netlist represented by a graph, which of the following is true for 4-input LUT mapping in a typical FPGA?
    a. The 4-input LUT is typically realized using a 16x1 SRAM unit.
    b. Any circuit subgraph with up to 4 input edges and 1 output edge can be mapped to a LUT irrespective of the number of vertices included therein.
    c. In SRAM-based LUTs, the function of the LUT cannot be changed by downloading appropriate bit patterns in the associated RAM locations.
    d. All of the above

    Answer: (a) & (b)

    The LUTs in FPGA chips are realized using 16x1 SRAM units and an LUT can realize any arbitrary 4 input and single output Boolean function. Thus option (a) and (b) are true. The functionality of a LUT can be changed by downloading appropriate bit stream in the SRAM; hence option (c) is false.

4. For which design style, the following statement is true?
    "Chip *fabrication cost consists of designing standard mask that is shared among several customers, followed by customization cost that is to be separately borne by every customer*."
    a. Gate array

b. FPGA
c. Standard cell
d. Full custom

Answer: (a)

A standard mask is designed in case of gate array based implementation, consisting of an array of unconnected transistors or gates. Depending on the function to be realized, the transistors and gates can be suitably interconnected (customization step). Thus option (a) is true.

FPGA is predesigned device that is needed to be programmed or configured whereas standard cell based design style use predesigned cell as building blocks and in full custom style the placement, orientation and geometry of each individual transistor has been considered. Thus options (b), (c) and (d) are false.

5. Which of the following is true for standard cell based design?
   (i) The heights of the cells are fixed but the widths can be different.
   (ii) Only fixed number of cells can be placed in a row.
   (iii) Over-the-cell routing is not possible.
   (iv) It requires less design effort as compared to full custom design.
   a. (i) and (iv)
   b. (ii) only
   c. (ii) and (iii)
   d. (iv) only

   Answer: (a)

   In case of standard cell based design, (ii) is false since in a row number of cells that can be placed is variable as cell width differs, and(iii) is also false as over-the-cell routing is also possible.

6. Which of the following represents the correct ordering with respect to speed of circuits (slowest to fastest)?
   a. Standard cell, FPGA, Full custom, Gate array
   b. FPGA, Gate array, Standard cell, Full custom
   c. Gate array, FPGA, Standard cell, Full custom
   d. None of the above

   Answer: (b)

   Here the correct ordering should be FPGA, Gate array, Standard cell, Full custom.

7. Which of the following statements are true?
   a. A test bench is required when we want to simulate a design.
   b. Technology specific mapping of a design does not require a test bench.
   c. Mapping the design to a field programmable gate array requires a test bench.
   d. Simulation means generation of a gate level netlist from a behavioral specification.

   Answer: (a) and (b)

Test bench is only required for simulating the behavior of a design, it has no role in case of technology specific mapping. Thus options (a) and (b) are true, and option (c) is false. Finally simulation means verifying the behavior of the design applyingcertain inputs, thus option (d) is false.

8. Which of the following is true about the following verilog module?

```
module guess (A, B, C, D, E, F, Y);
   input A,B,C,D,E,F;  output Y,
   wire t1, t2, y;
   nand #1 G1 (t1,A,B,C)
           G2 (t2,D,E,F);
   and  #1 G3 (y,t1,t2);
endmodule
```

   a. The module represents a design at the structural level.
   b. The modulerepresents a designatthe functional level.
   c. The module represents a design at the layout level.
   d. None of the above.

   Answer: (a)

   Since the module is designed using gates and interconnects, it represents the design at the structural level. Thus option (a) is true.

9. What function do the following Verilog module implement?

```
module guess (f, a, b, c);
   inputa,b,c;  output f,
   wire t1, t2;
   and#1 G1 (t1,a,b);
   nor   #1 G2 (t2,b,c);
   nand  #1 G3 (f,t1,t2);
endmodule
```

   a. f = a + b + c
   b. f = a' + b + c'
   c. f = 1
   d. None of the above

   Answer: (c)

   Here t1 = a.b, t2 = (b+c)' and

   f = [a.b.(b+c)']'

   = [a.b.b'.c']' //By DeMorgan's Law

   =[0]' //bb' = 0

   =1

10. What function do the following Verilog module implement?

```
module guess (f, a, b, c);
   inputa,b,c;  output f,
   wire t;
   assign t = (!a & b) | c;
   assign f = a ^ !t;
endmodule
```

   a. f = a'.b' + a'.b'.c

b. f = a.c + a'.b'.c'
c. f = a'.c + a'.b.c'
d. None of the above

Answer: (b)

Here t = a'.b + c and

f = a ⊕ (a'.b + c)'

= a.(a'.b+c) + a'.(a'.b+c)' // x ⊕ y = x'.y + x.y'

= a.a'.b + a.c + a'.(a'.b+c)' // By distributive law

= a.a'.b + a.c + a'.(a'.b)'.c' // By DeMorgan's law

= 0 + a.c + a'.(a+b').c' // x.x' = 0

= a.c + a'.a.c' + a'.b'.c' // By distributive law

= a.c + a'.b'.c' // x.x' = 0